Research Report on
Resource Requirements Research on the Quantum Threat Timeline

Prepared for:

Global Risk Institute
28 February 2017

evolutionQ

A resource estimation framework for quantum attacks against cryptographic functions

GRI quantum risk assessment report Sep. 2016 - Feb. 2017

Vlad Gheorghiu and Michele Mosca

evolutionQ Inc., Waterloo ON, Canada

28 February 2017

Abstract. We analyze the security of several symmetric-key cryptographic functions against an attack from a full-scale fault-tolerant quantum computer. We perform our analysis using an automated software framework. We also provide a set of lower bounds for attacks via Grover's algorithm on arbitrary symmetric cryptographic functions, as a function of their key sizes and physical error rate per gate, respectively.

1 Introduction

Cryptography is an indispensable ingredient in today's world, with applications ranging from secure communication to e-commerce and cryptographic currencies. Most of modern cryptography falls in one of two categories: *symmetric cryptography*, in which a common secret key is shared in advance among the parties, and the same key is used for both encryption and decryption, and *asymmetric cryptography*, or *public key cryptography*, in which a pair of public/secret keys is established among the parties with no need of prior secrecy whatsoever; most often the secret key is used for decryption and the public key is used for encryption, however there are schemes in which the roles are interchanged, such as in digital signatures. Some examples of symmetric ciphers are AES, DES, 3-DES, whereas examples of public key cryptographic protocols are RSA and Elliptic Curve Cryptography (ECC). Finally, a special category of cryptographic primitives consists of *hash functions*, which take an arbitrary long string and compress it (hash it) to a fixed size such that recovering the input from its hash is computationally unfeasible. Examples of hash functions currently in use are SHA1 (deprecated, but still heavily used), SHA-256, SHA-512, SHA3 and MD5.

Whereas symmetric key cryptography encrypts and decrypts the data exceedingly fast, it has the drawback that a secret key must be shared in advance. This fact poses a serious challenge in the real world, where communication channels cannot be considered a priori secure, such as in internet commerce. Public

key cryptography addresses this issue with a solution based on mathematical hardness of specific problems such as long integer factoring (RSA) or computing the discrete logarithm in a finite field (ECC). The drawback of public key cryptography is its relatively slow rate of encryption, which is often orders of magnitude slower than symmetric key schemes. For this reason most modern cryptographic protocols consist of a blend of symmetric cryptography, public key cryptography and hash functions. The public key protocol is used to establish a pair of public/private keys, which are further used to distribute a symmetric key, which is then used in a combination with a symmetric cypher as AES for the actual encryption and decryption. Hash functions are often used in combination with public key cryptography as part of *digital signatures* schemes such as DSA (Digital Signature Algorithm) or EC-DSA (Elliptic Curve Digital Signature Algorithm).

An important parameter that characterizes the strength of a cryptographic scheme is its *security parameter*, or, informally, the number of *bits of security*. We say that a scheme has a security parameter s, or equivalently, that it offers s bits of security against an adversary, provided that the best known attack by the adversary takes $T(s) = 2^s$ time steps. For example, RSA-2048 offers approximately 112 bits of security, ECC-256 offers 128 bits, whereas AES-128 offers 128 bits, AES-256 offers 256 bits, and SHA-256 offers 256 bits.

Quantum computing represents an entirely new model of computation, which harnesses the fundamental laws of quantum mechanics to perform computations. A number of quantum algorithms promise significant asymptotic speedups compared with their classical counterparts [1,2,3]. While most fields of research will be unaffected by these algorithms until large quantum computers are built, cryptography is affected by the possibility of these algorithms being run at any time in the future. The hardness assumptions underlying the public key cryptosystems currently in use – those related to factoring and variants of the discrete logarithm problem – are violated by quantum adversaries. Quantum Fourier sampling techniques break these cryptosystems in polynomial time [1,4]. As a result these cryptosystems can no longer be considered secure, and ultimately they will have to be replaced. Some standards bodies have already begun discussions about transitioning to new public key cryptographic primitives [5,6].

Symmetric primitives, by contrast, are weakened but not necessarily broken by quantum algorithms. The best generic attacks on symmetric primitives apply Grover's quantum search algorithm and achieve a corresponding quadratic improvement over exhaustive search in a black-box query model [2,7,8]. Such attacks are not formally efficient, but they do require a re-evaluation of the concrete security of symmetric primitives.

A conservative defense against attacks based on Grover's algorithm is to compensate for the potential square root loss in security by doubling the security parameter. This may mean doubling the key size for a cipher, or doubling the output length for a hash function. This is a suitable response for the cryptographer who wants to make worst case assumptions about the potential power of quantum computers. Others, however, may want to know either 1) the exact

cost of an attack based on Grover's algorithm for a particular parameterization of a cryptosystem, or 2) the minimal security parameter that provides "adequate protection" in the sense of [9,10,11].

Estimating either of these quantities requires close analysis of the cost of a realistic implementation of Grover's algorithm. Overhead is introduced at the logical level by the reversibility constraint on quantum computations and by the structure of the Grover iteration itself. Additional overhead may be introduced by fault-tolerance mechanisms required by a particular model of quantum computation.

In this report we present an estimate of the cost of performing realistic attacks on a variety of cryptographic schemes using a fault-tolerant quantum computer where the quantum error correction is performed using a *surface code*[12]. The latter seem to be the most promising candidate from the perspective of experimental realizability. We focus on symmetric schemes attacks via Grover's algorithm.

2 Methodology

2.1 Overview

We execute the following procedure for each cryptographic function. First, we implement the function as a reversible circuit. We then use a quantum circuit optimization tool, T-par [13], to minimize the circuit's T-count and T-depth. This is necessary because T gates are expensive in our chosen model of quantum computation. With the optimized circuit in hand we estimate the cost of executing Grover's algorithm on a surface code based quantum computer. Figure 1 gives a high-level description of Grover's algorithm. The algorithm makes $\lfloor \frac{\pi}{4} 2^{k/2} \rfloor$ calls to G, the *Grover iteration*. The Grover iteration has two subroutines. The first, U_g, implements the predicate $g : \{0,1\}^k \to \{0,1\}$ that maps x to 1 if and only if $f(x) = y$. Each call to U_g involves two calls to a reversible implementation of f and one call to a comparison circuit that checks whether $f(x) = y$.

Our resource estimates focus on the number of logical qubits in the fault-tolerant circuit and the overall depth of the circuit in units of surface code cycles. Each surface code cycle involves the execution of a classical syndrome decoding routine for every logical qubit. Thus in estimating these quantities we obtain the cost of an attack purely in terms of classical computing resources. Separately, we obtain an estimate for the number of physical qubits required for the circuit, and an estimate for the wall-clock time of the computation.

Our resource estimation methodology takes into account several of the layers between the high level description of an algorithm and the physical hardware required for its execution. Our approach is modular should assumptions about any of these layers change, and hence it allows one to calculate the impact of improvements in any particular layer. We illustrate our method schematically in Fig. 2.

All of the analysis in this report is performed via an automated modular software framework written by us in Python.

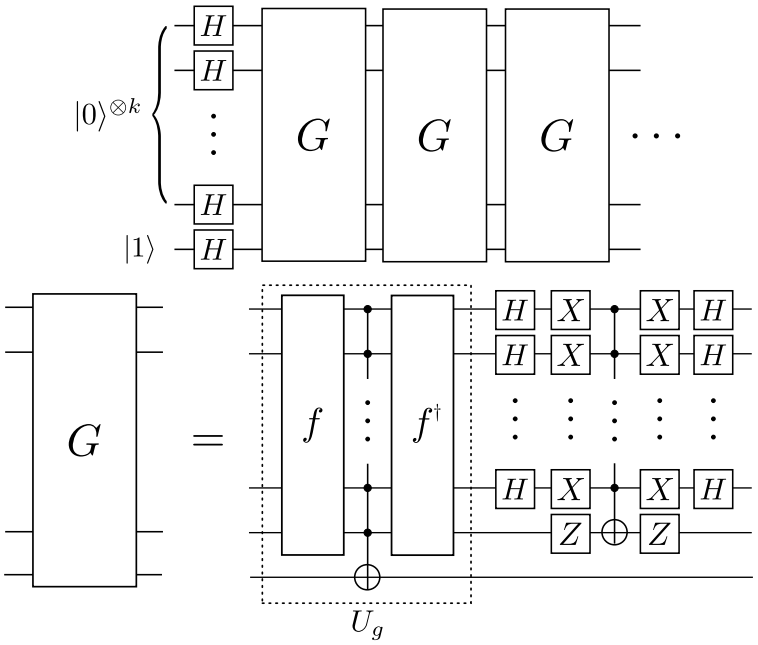

Fig. 1. Grover searching with an oracle for $f : \{0,1\}^k \to \{0,1\}^k$.

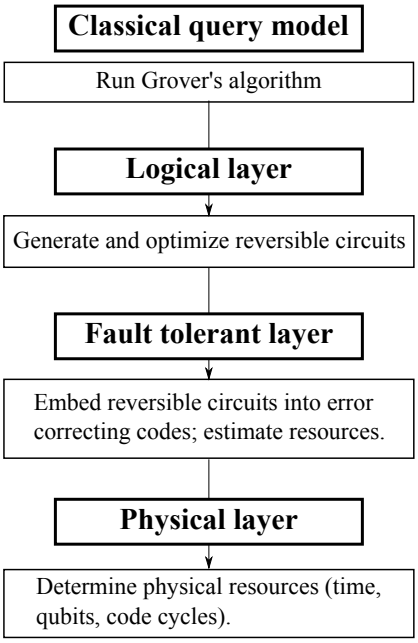

Fig. 2. Analyzing an attack against a symmetric cryptographic function with a fault-tolerant quantum adversary.

2.2 Cost metric for quantum computation

The majority of the overhead for quantum computation, under realistic assumptions about quantum computing architectures, comes from error detection and correction.

There are a number of error correction methods in the literature, however the most promising, from the perspective of experimental realizability, is the surface code [12]. Surface codes allow for the detection and correction of errors on a two-dimensional array of nearest-neighbor coupled physical qubits.

A distance d surface code encodes a single logical qubit into an $n \times n$ array of physical qubits ($n = 2d - 1$). A classical error correction algorithm must be run at regular intervals in order to prevent the propagation of physical qubit errors and, ultimately, to prevent logical errors. This algorithm is just part of the larger classical control infrastructure required to execute quantum algorithms. Every surface code *cycle* involves some number of one and two-qubit physical quantum gates, physical qubit measurements, and classical processing to detect and correct errors.

The need for classical processing allows us to make a partial comparison between the cost of classical and quantum algorithms for any classical cost metric. The fact that experts in quantum system engineering consider classical processing to be a bottleneck for quantum computation [14] suggests that an analysis of the classical processing may serve as a good proxy for an analysis of the cost of quantum computation itself.

Performing this analysis requires that we make a number of assumptions about how quantum computers will be built, not least of which is the assumption that quantum computers will require error correcting codes, and that the surface code will be the code of choice.

Assumption 1 *The resources required for any large quantum computation are well approximated by the resources required for that computation on a surface code based quantum computer.*

Fowler et al. [15] give an algorithm for the classical processing required by the surface code. A timing analysis of this algorithm was given in [14], and a parallel variant was presented in [16]. Under a number of physically motivated assumptions, the algorithm of [16] runs in constant time per round of error detection. It assumes a quantum computer architecture consisting of an $L \times L$ grid of logical qubits overlaid by a constant density mesh of classical computing units. More specifically, the proposed design involves one ASIC (application-specific integrated circuit) for each block of $C_a \times C_a$ physical qubits. These ASICs are capable of nearest-neighbor communication, and the number of rounds of communication between neighbors is bounded with respect to the error model. The number of ASICs scales linearly with the number of logical qubits, but the constant C_a, and the amount of computation each ASIC performs per time step, is independent of the number of logical qubits.

Each logical qubit is a square grid of $n \times n$ physical qubits where n depends on the length of the computation and the required level of error suppression.

We are able to estimate n directly. Following [14] we will assume that $C_a = n$. The number of classical computing units we estimate is therefore equal to the number of logical qubits in the circuit. Note that assuming $C_a = n$ introduces a dependence between C_a and the length of the computation, but we will ignore this detail. Since error correction must be performed on the time scale of hundreds of nanoseconds ($200ns$ in [12]), we do not expect it to be practical to make C_a much larger than n. Furthermore, while n depends on the length of the computation it will always lie in a fairly narrow range. A value of $n < 70$ is sufficient even for the extremely long computations we consider. The comparatively short modular exponentiation computations in [12] require $n > 31$. As long as it is not practical to take C_a much larger than 70, the assumption that $C_a = n$ will introduce only a small error in our analysis.

Assumption 2 *The classical error correction routine for the surface code on an $L \times L$ grid of logical qubits requires an $L \times L$ mesh of classical processors (i.e. $C_a = n$).*

The algorithm that each ASIC performs is non-trivial and estimating its exact runtime depends on the physical qubit error model. In [14] evidence was presented that the error correction algorithm requires $O(n^2)$ operations, on average, under a reasonable error model. This work considered a single qubit in isolation, and some additional overhead would be incurred by communication between ASICs. A heuristic argument is given in [16] that the communication overhead is also independent of L, i.e. that the radius of communication for each processor depends on the noise model but not on the number of logical qubits in the circuit.

Assumption 3 *Each ASIC performs a constant number of operations per surface code cycle.*

Finally we (arbitrarily) peg the cost of a surface code cycle to the cost of a cryptographic function invocation. If we assume, as in [12], that a surface code cycle time on the order of $100ns$ is achievable, then we are assuming that each logical qubit is equipped with an ASIC capable of evaluating several million function invocations per second.

Assumption 4 *The temporal cost of one surface code cycle is equal to the temporal cost of one cryptographic function invocation.*

Combining Assumptions 1, 2, and 4 we arrive at the following metric for comparing the costs of classical and quantum computations.

Cost Metric 1 *The cost of a quantum computation involving ℓ logical qubits for a duration of σ surface code cycles is equal to the cost of classically evaluating a cryptographic function $\ell \cdot \sigma$ times. Equivalently we will say that* one logical qubit cycle *is equivalent to* one cryptographic function invocation.

We will use the term "cost" to refer either to logical qubit cycles or to cryptographic function invocations.

2.3 Fault-tolerant cost

The T gate is the most expensive in terms of the resources needed for implementing a circuit fault-tolerantly in a surface code. Most known schemes implement the T gate using an auxiliary resource called a *magic state*. The latter is usually prepared in a faulty manner, and purified to the desired fidelity via a procedure called *magic state distillation*. Fault-tolerant magic state *distilleries* (circuits for performing magic state distillation) require a substantial number of logical qubits.

Let T_U^c denote the T-count of a circuit U (i.e., total number of logical T gates), and let T_U^d be the T-depth of the circuit. We denote by $T_U^w = T_U^c/T_U^d$ the T-width of the circuit (i.e., the number of logical T gates that can be done in parallel on average for each layer of depth). Each T gate requires one logical magic state of the form

$$|A_L\rangle := \frac{|0_L\rangle + \mathrm{e}^{\mathrm{i}\pi/4}\,|1_L\rangle}{\sqrt{2}} \tag{1}$$

for its implementation. For the entirety of U to run successfully, the magic states $|A_L\rangle$ have to be produced with an error rate no larger than $p_{out} = 1/T_U^c$.

The magic state distillation procedure is based on the following scheme. The procedure starts with a physical magic state prepared with some failure probability p_{in}. This faulty state is then *injected* into an error correcting code, and then by performing a suitable distillation procedure on the output carrier qubits of the encoded state a magic state with a smaller failure probability is distilled. If this failure probability is still larger than the desired p_{out}, the scheme uses another layer of distillation, i.e. concatenates the first layer of distillation with a second layer of distillation, and so forth. The failure probability thus decreases exponentially.

In the following, we use the Reed-Muller 15-to-1 distillation scheme introduced in [17]. Given a state injection error rate p_{in}, the output error rate after a layer of distillation can be made arbitrarily close to the ideal $p_{dist} = 35p_{in}^3$ provided we ignore the logical errors that may appear during the distillation procedure (those can be ignored if the distillation code uses logical qubits with high enough distance). As pointed out in [18] logical errors do not need to be fully eliminated. We also assume that the physical error rate per gate in the surface code, p_g, is approximately 10 times smaller than p_{in}, i.e. $p_g = p_{in}/10$, as during the state injection approximately 10 gates have to perform without a fault before error protection is available (see [19] for more details).

We define ε so that εp_{dist} represents the amount of logical error introduced, so $p_{out} = (1+\varepsilon)p_{dist}$. In the balanced case $\varepsilon = 1$ the logical circuit introduces the same amount of errors as distillation eliminates.

3 Results

In the following section we summarize our main results. For each cryptographic function we analyze we tabulate the resources needed to run the logical circuit

Algorithm 1 Estimating the required number of rounds of magic state distillation and the corresponding distances of the concatenated codes

Input: $\varepsilon, p_{in}, p_{out}, p_g (= p_{in}/10)$
$d \leftarrow$ empty list []
$p \leftarrow p_{out}$
$i \leftarrow 0$
repeat
    $i \leftarrow i + 1$
    $p_i \leftarrow p$
    Find minimum d_i such that $192 d_i (100 p_g)^{\frac{d_i+1}{2}} < \frac{\varepsilon p_i}{1+\varepsilon}$
    $p \leftarrow \sqrt[3]{p_i/(35(1+\varepsilon))}$
    d.append(d_i)
until $p > p_{in}$
Output: $d = [d_1, \ldots, d_i]$

(under "Parameters for the main circuit"), as well as the additional overhead introduced by the need of state distillation in the fault-tolerant layer (under "Parameters for distillation"). Finally we display the total cost in terms of surface code cycles, the corresponding security parameter s and the total wall time (in years) required to break the scheme (under "Costs"). The physical error rate per gate p_g is assumed to be of the order of 10^{-5}, a extremely optimistic number considered today's technology.

3.1 Hash functions - SHA-256 and SHA3

We first analyze 2 hash functions: SHA-256 and SHA3. Our results are summarized in Table 1 and Table 2, and are in accordance with [20]. The T-optimized logical circuits are taken from [20].

3.2 Symmetric ciphers - AES-128, AES-192 and AES-256

In this subsection we analyze 3 symmetric ciphers, namely AES-128, AES-192 and AES-256, respectively. The T-optimized logical circuits are taken from [21].

3.3 Intrinsic cost of Grover's algorithm with trivial oracles - lower bounds

In the ideal case (no error correction overhead), Grover's algorithm offers a quadratic speedup over classical searching. However, when the overhead introduced by fault-tolerance is taken into account, an additional polynomial overhead is introduced in Grover. Namely, $\Theta(2^{k/2})$ Grover iterations cost $\approx k^v 2^{k/2}$ logical qubit cycles for some real v independent of k. Then an adversary who is willing

Parameters for the main circuit	
T-count	1.27×10^{44}
T-depth	3.76×10^{43}
Logical qubits	2402
Surface code distance	43
Physical qubits	1.39×10^{7}
Parameters for distillation	
Logical qubits	3600
Surface code distance(s)	[33, 13, 7]
Magic state factories in parallel	1
States produced per factory	4
Physical qubits	5.51×10^{5}
Costs	
Surface code cycles	153.8
Total cost (security parameter)	**166.36**
Total walltime	1.26×10^{32} years

Table 1. SHA-256

Parameters for the main circuit	
T-count	2.71×10^{44}
T-depth	2.31×10^{41}
Logical qubits	3200
Surface code distance	44
Physical qubits	1.94×10^{7}
Parameters for distillation	
Logical qubits	3600
Surface code distance(s)	[33, 13, 7]
Magic state factories in parallel	294
States produced per factory	4
Physical qubits	1.62×10^{8}
Costs	
Surface code cycles	146.46
Total cost (security parameter)	**166.47**
Total walltime	7.75×10^{29} years

Table 2. SHA3

to execute an algorithm of cost 2^C can use Grover's algorithm to search a space of k bits provided that

$$k/2 + v \log_2(k) \leq C. \tag{2}$$

We define the *overhead* of the circuit as v and the *advantage* of the circuit as k/C. Note that if we view k as a function of v and C then for any fixed v we have $\lim_{C\to\infty} k(v, C)/C = 2$, i.e. asymptotically, Grover's algorithm provides a quadratic advantage over classical search. However, here we are interested in non-asymptotic advantages.

Parameters for the main circuit	
T-count	9.23×10^{25}
T-depth	1.47×10^{24}
Logical qubits	2953
Surface code distance	19
Physical qubits	3.33×10^{6}
Parameters for distillation	
Logical qubits	240
Surface code distance(s)	[19, 9]
Magic state factories in parallel	21
States produced per factory	3
Physical qubits	1.28×10^{6}
Costs	
Surface code cycles	88.41
Total cost (security parameter)	**101.37**
Total walltime	2.61×10^{12} years

Table 3. AES-128

Parameters for the main circuit	
T-count	4.50×10^{35}
T-depth	7.46×10^{33}
Logical qubits	4449
Surface code distance	28
Physical qubits	1.09×10^{7}
Parameters for distillation	
Logical qubits	3600
Surface code distance(s)	[27, 11, 5]
Magic state factories in parallel	21
States produced per factory	3
Physical qubits	5.91×10^{6}
Costs	
Surface code cycles	121.22
Total cost (security parameter)	**137.51**
Total walltime	1.97×10^{22} years

Table 4. AES-192

In the following we use our automated framework to compute the overhead introduced solely by the need of error correcting the Grover's algorithm with a searching space of size 2^k, considering a trivial oracle (i.e., the identity gate). We vary the key size k, ranging from 56 bits all the way to 512 bits, as well as the physical error rate per gate p_g, ranging from 10^{-3} all the way down to 10^{-7}, respectively. Note that a physical error rate of 10^{-4} is considered optimistic with today's technology. We tabulate the security parameter for each combination of key size/physical error rate. In the ideal case (no error correction needed), the

Parameters for the main circuit	
T-count	2.42×10^{45}
T-depth	7.00×10^{43}
Logical qubits	6681
Surface code distance	37
Physical qubits	2.86×10^{7}
Parameters for distillation	
Logical qubits	3600
Surface code distance(s)	[33, 13, 7]
Magic state factories in parallel	9
States produced per factory	4
Physical qubits	4.96×10^{6}
Costs	
Surface code cycles	154.66
Total cost (security parameter)	**169.91**
Total walltime	2.29×10^{32} years

Table 5. AES-256

security parameter is $k/2$, i.e. half the size of the key. Our results are summarized in Table 6.

$k(bits)/p_g$	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}
56	59.88	55.61	55.32		
112	90.04	88.84	85.78	85.37	84.01
128	98.05	97.36	94.16	93.63	92.88
168	119.04	118.41	117.43	114.6	114.18
192	134.74	130.72	130.2 0	126.99	126.16
256	167.25	163.51	163.48	162.08	159.61
384	231.72	231.47	228.62	228.18	227.35
512	296.44	296.83	293.41	292.1	292.7

Table 6. Security parameter as a function of the key size and physical error rate for Grover's algorithm with a trivial oracle. Here k denotes the key size (or, equivalently, 2^k is the size of the searching space) and p_g represents the physical error rate per gate.

One can easily observe that $k = 56$ represents the "sweet spot", i.e. the region where a quantum computer ceases to offer any advantage whatsoever compared to a classical brute force searching on current silicon-based computers. In fact, if the physical error rate per gate is 10^{-3} or larger, the overhead introduced by error correcting the Grover's algorithm itself is greater than any advantage whatsoever. We conclude that for small key sizes such as $k = 56$ and lower, it is more advantageous to use classical brute force searching than employing a fault-tolerant quantum computer.

The results in Table 6 represent relatively strong lower bounds for the security parameter of any symmetric cryptographic primitive against a fault-tolerant quantum adversary, as the latter have non-trivial oracles which introduce additional overhead. The only assumption we made is the use of a surface code as the primary means of quantum error correction. We cannot say for certainty that topological error correction is the best scheme, perhaps in the future other more resource-efficient schemes will be discovered. However, as of today, the surface code is by far the most promising approach to achieving large scale fault-tolerance.

4 Future directions

In the near future we aim to extend our analysis to a variety of asymmetric (public key) cryptographic schemes, most notably to RSA with different modules and Elliptic Curve Cryptography over various curves, as well as to digital signature schemes over those primitives.

References

1. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal on Computing 26(5), 1484–1509 (1997), http://link.aip.org/link/?SMJ/26/1484/1
2. Grover, L.K.: Quantum mechanics helps in searching for a needle in a haystack. Phys. Rev. Lett. 79, 325–328 (Jul 1997), http://link.aps.org/doi/10.1103/PhysRevLett.79.325
3. Jordan, S.: Quantum Algorithm Zoo (February 2016), http://math.nist.gov/quantum/zoo/
4. Boneh, D., Lipton, R.J.: Quantum Cryptanalysis of Hidden Linear Functions. In: Coppersmith, D. (ed.) Advances in Cryptology - CRYPTO'95, pp. 424–437. No. 963 in Lecture Notes in Computer Science, Springer Berlin Heidelberg (Aug 1995)
5. Agency, U.S.N.S.: NSA Suite B Cryptography - NSA/CSS. NSA website (August 2015), https://www.nsa.gov/ia/programs/suiteb_cryptography/
6. Chen, L., Jordan, S., Liu, Y.K., Moody, D., Peralta, R., Perlner, R., Smith-Tone, D.: Report on post-quantum cryptography. National Institute of Standards and Technology Internal Report 8105 (February 2016)
7. Boyer, M., Brassard, G., Høyer, P., Tapp, A.: Tight bounds on quantum searching. Fortschritte der Physik 46(4-5), 493–505 (1998), http://dx.doi.org/10.1002/(SICI)1521-3978(199806)46:4/5<493::AID-PROP493>3.0.CO;2-P
8. Gilles, B., Peter, H., Michele, M., Alain, T.: Quantum amplitude amplification and estimation. Quantum Computation and Quantum Information, Samuel J. Lomonaco, Jr. (editor), AMS Contemporary Mathematics (305), 53–74 (2002), e-print arXiv:quant-ph/0005055
9. Lenstra, A.K.: Handbook of Information Security, chap. Key Lengths. Wiley (2004)
10. Lenstra, A.K., Verheul, E.R.: Selecting cryptographic key sizes. Journal of Cryptology 14(4), 255–293 (Aug 2001)

11. Blaze, M., Diffie, W., Rivest, R., Schneier, B., Shimomura, T., Thompson, E., Weiner, M.: Minimal key lengths for symmetric ciphers to provide adequate commercial security. Tech. rep., An ad hoc group of cryptographers and computer scientists (1996)
12. Fowler, A.G., Mariantoni, M., Martinis, J.M., Cleland, A.N.: Surface codes: Towards practical large-scale quantum computation. Physical Review A 86(3), 032324 (Sep 2012), http://link.aps.org/doi/10.1103/PhysRevA.86.032324
13. Amy, M., Maslov, D., Mosca, M.: Polynomial-time t-depth optimization of Clifford+T circuits via matroid partitioning. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 33(10), 1476–1489 (Oct 2014)
14. Fowler, A.G., Whiteside, A.C., Hollenberg, L.C.L.: Towards practical classical processing for the surface code: Timing analysis. Physical Review A 86(4), 042313 (Oct 2012), http://link.aps.org/doi/10.1103/PhysRevA.86.042313
15. Fowler, A.G., Whiteside, A.C., Hollenberg, L.C.L.: Towards Practical Classical Processing for the Surface Code. Physical Review Letters 108(18), 180501 (May 2012), http://link.aps.org/doi/10.1103/PhysRevLett.108.180501
16. Fowler, A.G.: Minimum weight perfect matching of fault-tolerant topological quantum error correction in average $O(1)$ parallel time. arXiv:1307.1740 [quant-ph] (Jul 2013), http://arxiv.org/abs/1307.1740, arXiv: 1307.1740
17. Bravyi, S., Kitaev, A.: Universal quantum computation with ideal Clifford gates and noisy ancillas. Phys. Rev. A 71, 022316 (Feb 2005), http://link.aps.org/doi/10.1103/PhysRevA.71.022316
18. Fowler, A.G., Devitt, S.J., Jones, C.: Surface code implementation of block code state distillation. Scientific Reports 3, 1939 EP – (06 2013), http://dx.doi.org/10.1038/srep01939
19. Fowler, A.G., Mariantoni, M., Martinis, J.M., Cleland, A.N.: Surface codes: Towards practical large-scale quantum computation. Phys. Rev. A 86, 032324 (Sep 2012), http://link.aps.org/doi/10.1103/PhysRevA.86.032324
20. Amy, M., Matteo, O.D., Gheorghiu, V., Mosca, M., Parent, A., Schanck, J.: Estimating the cost of generic quantum pre-image attacks on SHA-2 and SHA-3 (2016), arXiv:1603.09383v1
21. Grassl, M., Langenberg, B., Roetteler, M., Steinwandt, R.: Applying Grover's algorithm to AES: quantum resource estimates, e-print arXiv:1512.04965 [quant-ph]